UAV Avionics "Hardware in the Loop" Simulator

McManus, I.A.*, Greer, D. G. & Walker, Dr. R. A.
Cooperative Research Centre for Satellite Systems
Queensland University of Technology, Brisbane

Abstract

The avionics systems, especially the flight management and control systems, are some of the most critical elements of any aircraft, more so for an autonomous aircraft. Testing these systems in a realistic synthetic environment is an important process. It is logical therefore, to develop a suitable simulation environment in which the avionics systems can be tested. It is desirable for this system to be easy to implement, cost effective and representative of the aircraft being simulated.

This paper discusses the development of such a simulation system, known as the Aircraft Simulation And Testing Environment (ASATE). ASATE is the continuation of work initiated in 2001 for the testing of avionics developed as part of the QUT UAV project. ASATE incorporates commercially available software packages including X-Plane for the flight visualisation and Matlab Simulink for the implementation of the dynamic flight model, sensor models and control algorithms. The mission management computer and other systems including the flight control system, are connected to the Simulink/X-Plane systems via ASATE. ASATE provides the means for designers to construct mission and control algorithms and to test the avionics hardware and software systems in real time, by simulating full missions, with realistic visual feedback of the aircraft's response.

ASATE has been successfully employed to verify the flight management system and the flight control system of the QUT UAV. A set of control system tutorials, based on ASATE, have also been developed to assist in the teaching of flight control systems at QUT. This paper describes the development of ASATE and its usage for simulating aircraft missions (excluding takeoff and landing) and as an educational tool for students.

1. Introduction

Research is currently underway at QUT on the area of intelligent avionics systems to facilitate low cost UAV deployment in civilian airspace. The research is focusing on creating an intelligent agent based system which will replicate the functions performed by a human pilot. Currently no requirements have been developed for the certification of UAVs. The goal of the research is to develop an architecture which provides a potential solution to this problem of UAV certification.

An important aspect of the research will be the proving of the capability of the intelligent agent system to sufficiently replicate the functions of a pilot. Proving this capability requires two components. First, test plans need to be developed for the intelligent agent system based upon consultations with CASA, pilots and UAV operators. Second, a simulation system is required in which these test plans can be executed. The focus of this paper is on the development of a simulation system to test the intelligent agent based system being developed at QUT.

Simulated testing assists in the identification and elimination of problems in the hardware and software of these systems. By making the simulation environment realistic and extensive it is possible to simulate a wide range of scenarios. The more scenarios capable of being simulated the more valuable the testing which can be performed using the simulation environment. The combination of a capable simulation system and well structured test plans provides the capability to test all elements of a system from the highest level to the lowest.

The process of testing provides benefits other than just fault detection and elimination. Using results from simulations, an operational profile of the system can be constructed. Based upon this profile, elements of the system with poor performance can be optimised. More complicated failures, additional test plans and plans for future expansion can also be derived from this profile of the system's operation.

An essential component of the simulation system is the incorporation of the actual flight hardware into the simulation loop. Incorporating the flight hardware makes the simulation system more complex and potentially more expensive however the benefits from this inclusion are significant. Simply linking together software and using simulated hardware will not identify all of the potential failures. Personal

experience has shown that the philosophy of "if the code works on the simulator it will work on the hardware" is a myth. Complex interactions frequently occur between hardware and software. These interactions are compounded when the system involved contains multiple interlinked systems as in the case of an aircraft.

It is this complexity, present in complex hardware and software systems, which makes the simulated testing of systems with the actual hardware in the loop essential. By fully exercising the system hardware and software, through the simulated execution of missions, it is possible to identify more potential problems, than by simply testing the hardware and software in isolation.

A solution to this problem of hardware in the loop based simulated testing was first developed at QUT in 2001 by James Hutchins and Iain McManus [1, 2]. This system was the Aircraft Simulation And Testing Environment (ASATE). ASATE linked the flight hardware developed for the QUT Unmanned Airborne Vehicle (QUAV) project to the X-Plane flight simulation system.

The ASATE system has evolved through three additional iterations since this first version. With each iteration the modularity and capability of the system has been increased. The third iteration of ASATE was developed in 2002 for the purpose of teaching students about control systems. This version of ASATE was also used for testing of flight control algorithms developed for the QUAV project in 2002. The most recent (4th iteration) of ASATE makes use of sensor and flight models independent of the visualisation system and introduces a number of optimisations to remove deficiencies present in previous generations.

This paper discusses the architecture and components of the current iteration of the ASATE system. The applications to which ASATE has been, is being and will be applied, to are also presented. The ASATE system has the following capabilities.

- Customisable sensor and aircraft models
- Ability to simulate the injection of hard and soft failures in sensors and the aircraft itself
- Ability to simulate the correction of failures
- Incorporates flight hardware in the simulation loop
- High quality flight visualisation system

2. *System Architecture*

The architecture of the current generation of the ASATE system is shown in Figure 1 below.

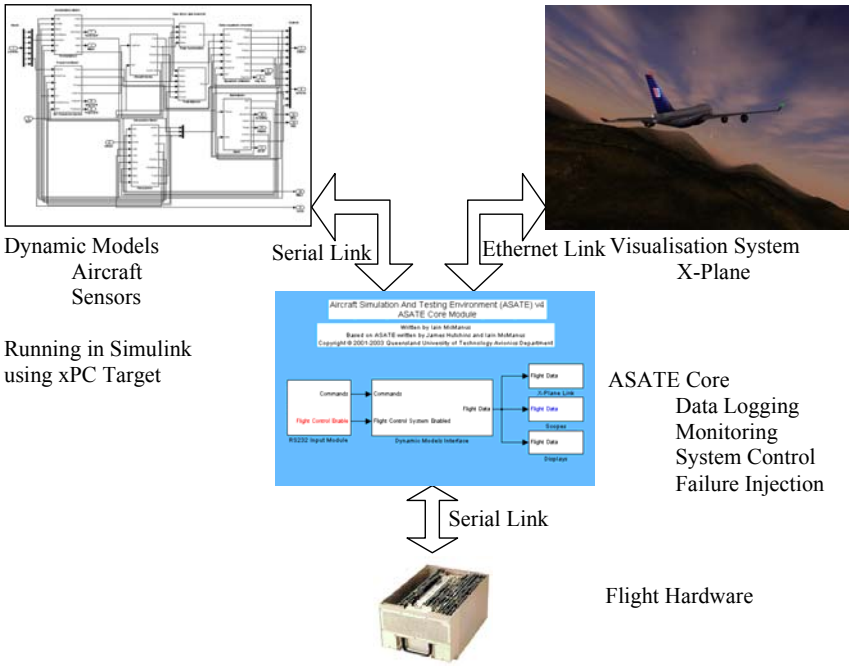

Figure 1. ASATE System Architecture

The ASATE system is comprised of three separate systems all linked together via the ASATE Core. The ASATE Core, flight and sensor models and the visualisation system run on three separate

PC based systems. An Ethernet link is used for communications between the ASATE core and the flight visualisation system. Serial communications are used between the core and the flight and sensor models. The interface between the core and the flight hardware is currently handled through serial communications however this is easily modifiable.

More details on the components of the ASATE system are provided in the proceeding sections.

3. ASATE Core

The purpose of the ASATE Core is to link together the visualisation system, the sensor and flight models and the flight hardware. The core itself is a model running in Matlab Simulink with custom blocks which enable it to communicate via serial and ethernet communications. The serial communications are used to communicate with the flight hardware and the sensor and flight models running on the embedded system. The ethernet communications are used to send commands to the flight visualisation system.

The need for the ASATE Core can be eliminated by having the embedded system talk directly to X-Plane and the flight hardware. However the current architecture (with the ASATE Core) is advantageous for a number of reasons.

Firstly, it provides a central hub where all system data is available making data logging a simple and easy task. Secondly, this approach provides a greater level of modularity to the system. With this architecture it is a simple matter to use a different visualisation system, sensor and flight model or different flight hardware. The interfaces are easily adaptable and changing one part of the system does not require every element of the system to be changed. Thirdly, the hub based architecture is easier to implement and maintain. Switching to a different visualisation system is simply a matter of changing one element of the core. Major changes to the system do not need to be made, only the interface component of the system is changed. Finally, it provides a central point for controlling the simulation. Through the core all elements of the system can be monitored and controlled.

The ASATE Core is responsible for: passing commands from the flight hardware to the dynamic models; passing aircraft state information to the visualisation system and the flight hardware; monitoring and controlling the dynamic algorithms; and, logging all data which passes through the core. The ASATE core is shown in Figure 2.

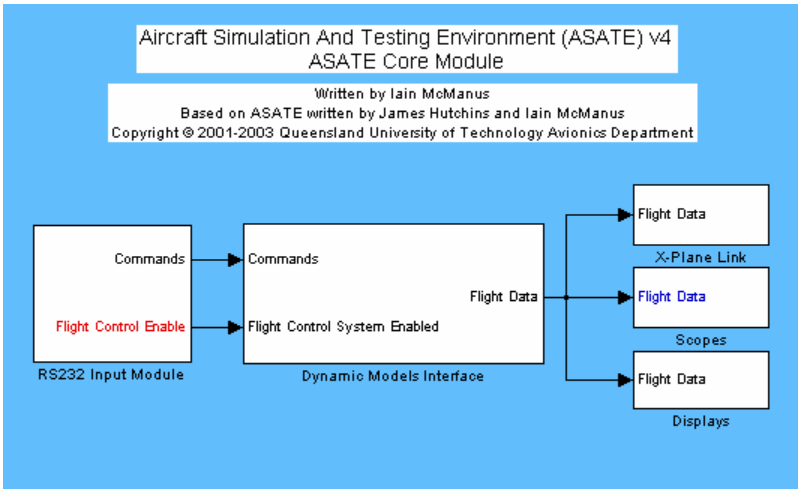

Figure 2. ASATE System Core

Matlab Simulink was chosen for the ASATE core (and the dynamic models) for a number of reasons. Simulink provides a simple and powerful environment for the implementation of control algorithms, and dynamic models. It is a simple matter to create add-ons for Simulink which expand its capabilities by providing additional interfaces (serial communications, ethernet etc). Simulink also provides a wide range of functions for logging, manipulating and viewing data. The final advantage of Simulink is that it is possible (using xPC Target) to compile a Simulink model and run it on any PC compatible system providing real time execution of the model. Simulink has also been used by other simulation systems designed for use in development and testing of control systems [3, 4].

4. Flight and Sensor Models

Previous versions of the ASATE system have relied upon the X-Plane flight simulation software for both the visualisation and modeling of the flight and sensor dynamics. The current version of the ASATE system uses a separate computer to handle the flight and sensor models. The flight model component is shown in Figure 3.

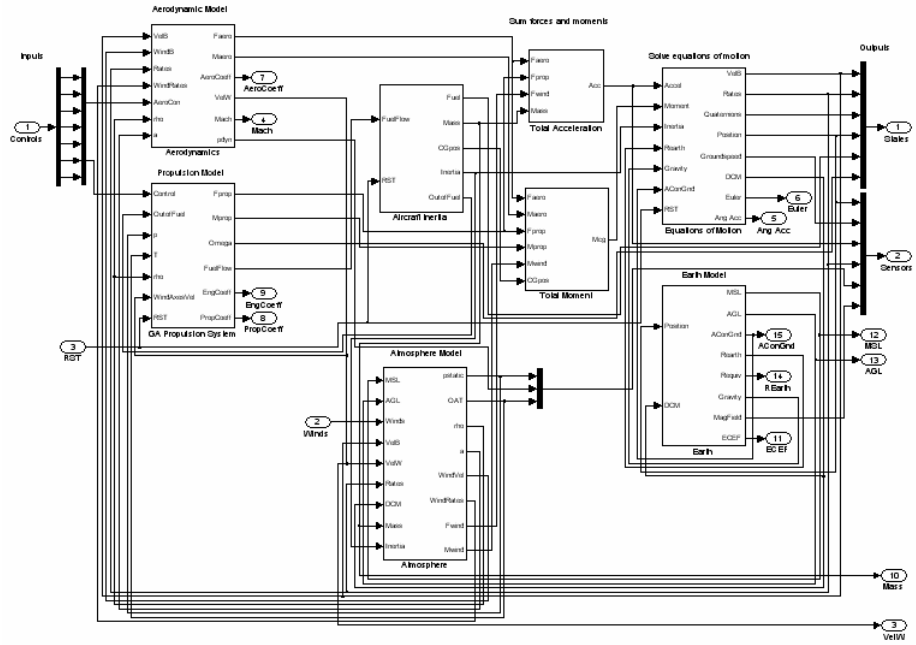

Figure 3. ASATE AeroSim Based Flight Model

This approach has a number of advantages over the previous approach of using the sensor and flight models provided by X-Plane.

Firstly, the previous versions of ASATE suffered from a major problem in that the Simulink based control systems were operating at a different rate to the X-Plane dynamic models. This meant that control systems designed using ASATE were not suitable for actual implementation on a real system. The control systems required a number of scaling factors to compensate for this lack of synchronisation. The current system implements the dynamic models on an embedded system using xPC Target (a component of Matlab Simulink's Real Time Workshop). This implementation ensures that the dynamic models are run in real time and provides a means to synchronise all elements of the ASATE system.

Secondly, although X-Plane provides the means to cause failures to systems it is not possible to simulate a wide range of failure types. For example fuel leakage, internal communications failure etc. The current system makes it possible to fully control every element of the simulated aircraft. A wide range of failures can be simulated and any repair actions can be simulated.

Thirdly, X-Plane provides only limited control over the flight and sensor models implemented within X-Plane. The flight models are essentially just an input/output block which is provided with the control deflections and then changes the aircraft state. No control over the sensor models is possible. This severely limited control makes it difficult to inject a wide range of failures into the system. The current system implements its own flight and sensor models providing complete control over every aspect of the system.

Overall, although the shift of the dynamic models increased the complexity and computing requirements of ASATE, the shift has made it possible to more fully simulate missions and removed a number of the problems and deficiencies present in earlier versions.

Currently the dynamic models of the aircraft and sensors are implemented in Matlab Simulink using the AeroSim blockset, freely available to academic institutions [5]. The AeroSim model provides a full nonlinear, six degrees of freedom aircraft model based upon the stability coefficients for an aircraft. AeroSim also provides standard sensor models. These models are implemented in such a

manner that they can easily be replaced with models based upon analyses of the sensors to be used on the actual aircraft.

This Simulink model is executed on a separate PC using the xPC Target system which compiles the model into executable code. This has the advantages of isolating the dynamic models from the remainder of the system and, as it executes in real time, it provides synchronisation for the other elements of ASATE.

The embedded Simulink model communicates to the ASATE core through a standard RS232 serial communications link. The aircraft states generated by the dynamic models are sent to the ASATE core to be sent on to the visualisation system and the flight hardware. Control deflections are received from the flight hardware via the core and used as the input to the dynamic models. Information about the current state of the dynamic models is provided to the core and commands to change aspects of the dynamic models can be sent by the core.

5. Visualisation System

Throughout all of its versions ASATE has always made use of the X-Plane flight simulation system for either flight visualisation or for the dynamic models. X-Plane is a commercially available flight simulation package produced by Laminar Research. X-Plane has been used throughout all of the versions of ASATE as it provides a simple interface for sending commands to X-Plane and receiving data back. Using standard ethernet communications it is possible to both send commands to X-Plane's flight model and to completely override the flight model and directly set the aircraft state. The X-Plane visualisation system is shown in Figure 4.

Figure 4. X-Plane Flight Visualisation System

Previous versions of the ASATE system used the dynamic models built into X-Plane for simplicity. When being used in this mode X-Plane was sent control surface commands and the flight data generated by X-Plane was received and passed to the control algorithms. The current ASATE system does not use the X-Plane dynamic models. In this mode aircraft state commands (position and orientation) are sent to X-Plane and no data is received back.

Commands are sent to X-Plane using a custom Simulink block in the ASATE core. This Simulink block was written specifically to communicate to X-Plane, however it could easily be adapted to communicate with any ethernet capable simulation system. The block is provided with the desired aircraft position and orientation which are then sent to X-Plane.

Although X-Plane is currently used and has been used in all previous versions the current architecture of ASATE means that the system is not dependent upon X-Plane. X-Plane is in fact not even required with the current architecture. It is provided simply as a means to view what the aircraft is

doing, it serves no functional purpose within the ASATE system. The replacement of X-Plane with a different visualisation system is simply a matter of replacing the current Simulink block which performs the communications with X-Plane. The AeroSim blockset, used for the dynamic models, already provides blocks to communicate with both the Flight Gear simulation system and Microsoft Flight Sim. The X-Plane communications block was designed to be compatible with these blocks and as such uses the same data structure and formatting as the AeroSim blocks.

6. Incorporation of Flight Hardware

The primary purpose of ASATE is to provide a simulation environment for the testing of the intelligent agent system being developed at QUT. The intelligent agent system will be tested while running on the flight hardware. In order to achieve this the flight hardware must be linked into the simulation system in as transparent a manner as possible. By this we mean that the flight hardware should ideally not be capable of detecting any difference between the simulation and the real world.

In practice achieving this level of transparency is both difficult and expensive. It means simulating a potentially large number of interfaces and has the negative aspect that it produces a simulation system tied to a specific set of hardware. The addition of new flight hardware means redesigning elements of ASATE and potentially the development of additional interface hardware. Both of which involve additional cost, time and complexity. This goes against the philosophy of making ASATE as simple, modular and hardware independent as possible.

The alternative approach is to implement a standard interface to which the flight hardware must conform. The onus is then place upon the developer of the flight hardware to ensure its design is compatible with this standardised interface. Additionally the connection to the simulation environment must occur as far upstream in the flight hardware as possible. This minimises the proportion of the flight hardware which it is not possible for ASATE to test. The incorporation of the flight hardware into the simulation system requires modifications to the flight hardware. An additional module must be added to the flight hardware to interface to the simulation environment.

All of the previous versions of ASATE used RS232 serial communications to link the flight hardware to the X-Plane dynamic models. However previous versions have not connected directly to the core of the ASATE system. Instead a separate program has been used to sit between the core of the ASATE system and the flight hardware. This simplified the implementation of the system however it had significant disadvantages in that it increased the complexity and the lag present in the system. Previous interfaces were also very specific to the hardware being linked in. This meant that hardware changes required significant modification to the system.

The current ASATE system also uses RS232 serial communications between the flight hardware and the ASATE core. This communications method was retained as serial communications are present on the majority of PC's and microprocessor based systems. All communications through this link are packet based and are either commands or reports. Reports are used by the ASATE core to provide sensor data to the flight hardware. Commands are sent from the flight hardware to provide new input to the dynamic model or to reconfigure aspects of the dynamic model. This implementation provides a more direct and flexible link into the ASATE system than previous approaches by using a custom Simulink block.

The disadvantage of the RS232 communication links are that they create potential bottlenecks in the simulation environment. The RS232 links can handle a maximum data throughput of 115, 200 bits per second (bps). Due to the large quantity of data which will be being transferred through these links (eg. aircraft state information) a slow down may occur due to the inability of the link to handle the full data throughput. This problem can be removed by changing the link between the ASATE core and the dynamic modules to ethernet (a 10+ Mbps link supported natively by xPC Target). Similarly, the link between the flight hardware and the ASATE core can be shifted to CAN (1 Mbps link) or a similar databus.

The serial communications are handled on the ASATE core side using a custom Simulink block similar to the block developed to interface to the X-Plane visualisation system. This block has been structured in such a way as to make the addition or modification of reports and commands a simple process.

The serial communications interface on the flight hardware side has not been designed as part of the ASATE system. This has been deliberately done to maintain the hardware independence of ASATE. However, a hardware interface compatible with both the ASATE system and the flight

hardware used on the QUAV project has been developed previously. This interface acted as a bridge between the ASATE system and the actual flight hardware. The interface was capable of receiving reports from ASATE which were then rebroadcast via a separate databus in the appropriate packet format. Similarly commands from the flight hardware were received, translated and then passed to the ASATE core. The advantage of this interface was that as far as the flight hardware was concerned there was no difference between the simulated world and the real world.

7. Applications and Results

ASATE has been an important part of many previous projects and is an active part of a number of current projects. The first two versions of ASATE were used to test and to demonstrate flight hardware and software developed for the QUAV project. The first two versions were capable of being linked to the hardware developed in 2001 for the QUAV project. This setup provided virtually transparent operation for the flight hardware with their being only minor differences between the simulated world and the real world in which the hardware would operate. ASATE was used to test the flight control, flight management and ground station components on the QUAV project in 2001. However this testing did not simulate full missions, only the actual in flight portion of the mission. Elements of the first versions of ASATE were also used by the QUT Flight Simulator project to handle the communications with the X-Plane simulation software.

The testing performed on the flight hardware and software using ASATE proved invaluable. Through this testing a number of previously unidentified bugs (mainly in interfaces) were identified and corrected. ASATE also proved to be a highly useful tool for developing control algorithms. Using ASATE control algorithms were tuned with feedback being provided in real time. This reduced the amount of time and work required to develop suitable control algorithms. The advantages of ASATE for the purpose of control system design led to the development of the third version of ASATE.

The third version of ASATE was developed for the purpose of providing a system for teaching undergraduate students about flight control systems. Using ASATE the students were able to design and test their control systems in real time receiving immediate visual feedback from any changes made to the controllers. The tutorials developed led the students through the development of lateral and longitudinal stability augmentation systems and control systems. Students are shown a series of different implementations of control systems and get to see exactly how the aircraft responds under these situations. This serves to teach the students to consider aspects of control design (eg. limits etc) which they otherwise would not have thought of. These tutorials will be used by second and third year Aerospace Avionics students in 2003 and were used by a small number of students in 2002. An example of the output generated by ASATE is shown in Figure 5. This output shows a plot of the aircraft altitude (in feet) versus the simulation time (in seconds). The plot shows the altitude profile of the aircraft undergoing this sequence: takeoff, capture and hold 1000 feet, capture and hold 5000 feet.

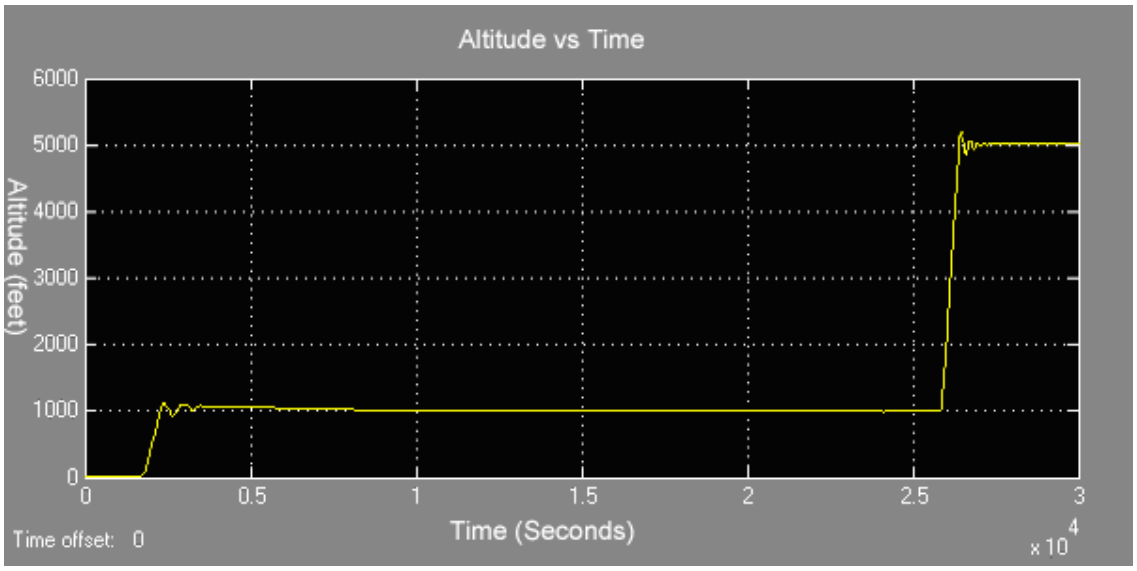

Figure 5. Altitude vs Time Plot Generated By ASATE

Although not specifically designed for it, the third version of ASATE was used for testing the flight controllers being developed for the QUAV Piper Cub in 2002. A number of mission scenarios were run with varying weather conditions and controller gains. Minor modifications were made near the end of 2002 to link the third version of ASATE to the current flight hardware being used on the QUAV project. This modified version was used to test and to demonstrate the groundstation hardware and software designed for the QUAV Piper Cub.

The fourth version of ASATE has been designed with hardware in the loop testing a priority. This system has been developed in order to test intelligent avionics systems currently being developed at QUT [6]. At QUT intelligent avionics systems are currently being developed for usage on unmanned aircraft. ASATE is a critical component of the development and testing of these systems and will be used extensively in the design process.

The intelligent avionics systems being developed at QUT will perform onboard mission planning and execution and failure detection and compensation. The current (fourth version) of ASATE will be used to simulate all aspects of the mission from takeoff through to landing. Faults will be injected into the system and then repaired by the intelligent avionics. This requires a level of realism in the simulation which only the current version of ASATE is capable of providing.

The usage of ASATE for simulation of vehicles other than aircraft (eg. satellites) has also been considered. With the current, and more modular, architecture there is no limitation on the situations to which ASATE could be applied. Providing a capable visualisation system is present, if visualisation is desired, then ASATE could be used to simulate satellite, ground based or underwater based operations. Minor modifications would be required to elements of the ASATE core and the dynamic models would need to be replaced. However the modular architecture of ASATE makes this a relatively simple task.

8. Problems Encountered

A range of problems were encountered in the development of the ASATE simulation environment, these problems were primarily related to the X-Plane simulation software. The first and most influential problem was due to the flight and sensor models provided by X-Plane. Recent versions of X-Plane permit failures to be injected into particular systems (eg. left aileron) however the levels of failure which can be injected are limited by X-Plane. For example it is not possible to put varying amounts of drift onto navigational sensors, the sensors are either working or not working. This problem meant that it was necessary to implement flight and sensor models, external to X-Plane, which were capable of simulating a configurable range of failures. As a result X-Plane is solely used to provide visual feedback of the aircraft state.

The second problem with X-Plane relates to the data format of the X-Plane communications. This format changes, sometimes significantly, with each new version of X-Plane. New versions of X-Plane typically come out every one to two months. This means that by designing ASATE to interface to one version of X-Plane it will not be able to interface to other versions of X-Plane without potentially major changes. This problem was solved by developing a database driven interface to X-Plane. A database defines the details for every X-Plane version and a generic software module uses this database to decode/encode the X-Plane communications. As a result when a new version of X-Plane is produced all that is required is to add the details for the new version into the database. ASATE is therefore capable of working with all current and, unless the data format changes completely, future versions.

The third problem relates to the coordinate system used by X-Plane. It is not possible in X-Plane to specify the location of an aircraft using latitude, longitude and altitude. Instead the location must be specified using the OpenGL coordinate system. OpenGL is the graphics system used by X-Plane. Because of the different coordinate systems a transformation must be performed to shift from latitude, longitude and altitude to the X-Plane coordinate system.

The final problem with X-Plane is of a cosmetic nature. When the flight model is overridden in X-Plane the graphics produced by X-Plane have a tendency to jitter. The amount of jitter varies according to the view (eg. outside from above, outside from the side, inside etc). The jitter is minor and can be corrected by switching to a different view, rotating the view or zooming in/out.

9. Discussion

Developing certification requirements for UAVs is essential. Rapid growth is occurring in the UAV industry with 12 applications for operators certificates having been made to CASA since June 2002.

UAV certification requirements currently do not exist. Research being conducted at QUT is developing an avionics architecture for low cost UAVs as a potential architecture for certification. This architecture is based upon incorporating the intelligence of a pilot into the UAV.

Certification requirements are only half of the solution. A systematic method is required to verify compliance with the certification requirements. This verification needs to be performed at two levels. Firstly, at the low level operation of the software. It must be proved, through the application of a standard such as DO-178B, that the software is robust and reliable. Secondly, the high level operation of the software needs to be verified. This involves proving that the UAV has been imparted with sufficient intelligence to allow it to operate safely in civilian airspace.

Proving that UAV has sufficient intelligence requires a set of test scenarios to which the system must be exposed. The test scenarios need to verify the correct response of the UAV avionics to anomalous events. In order to execute these test scenarios a simulation environment is required to perform the testing.

Simulated testing of aircraft avionics systems is already being performed extensively. The reasons for this are both simple and obvious. Simulated testing is a powerful tool for finding and eliminating system faults. Through simulated testing it is possible to uncover bugs not detected using other testing techniques. This has been discovered experimentally through the usage of the ASATE simulation system to test the avionics systems developed for the QUAV project.

The viability and potential of a simulation system are increased significantly when the simulation system is linked directly to the actual flight hardware. Using this approach it is possible to identify a wide range of faults (eg. interface problems) not otherwise detectable. Again this was proven experimentally through the usage of ASATE to test flight hardware on the QUAV project.

The capability to simulate full missions from takeoff through to landing only serves to further increase the power of the simulation system. With this capability the simulation system is capable of testing the flight hardware from the lowest level all the way through to the highest level.

10. Conclusions and Recommendations

The ASATE system has proven itself to be a highly capable and cost effective simulation environment. The effective cost to the university has been minimal as the necessary hardware and software equipment was already available. Usage of ASATE over the last two years has demonstrated its capabilities and the benefits gained from using a simulation system such as ASATE. ASATE has helped to detect faults which had not been discovered by previous system testing. It has been used to assess the performance of flight hardware and to optimise and modify the hardware. During its usage ASATE was also found to be a superb tool for usage in the design of control systems as it provided real time feedback of the effect of any modifications to the controller. This aspect of ASATE was further expanded and applied to the teaching field with the development of a set of control system based tutorials for students [7, 8]. These tutorials will provide an important learning tool for students due to the ability to actually see the effects of the control system.

This extensive and varied usage of ASATE has led to the most recent developments on the system which have made ASATE capable of simulating highly complex situations. This provides ASATE with the capability to provide a simulation environment for the next generation of avionics systems being developed at QUT.

Although the current version of the ASATE system is capable of meeting the current and near future needs for the QUAV project there are a number of avenues of expansion available. Firstly the visualisation system can be upgraded to accommodate a much wider range of simulations. A visualisation system capable of being used for aircraft and other vehicle mission simulations would provide great benefit. The logical companion to this expansion would be the development of dynamic algorithms for the simulation of satellites. In this way the capabilities of ASATE can be employed for the development and testing of satellite systems, and also for teaching purposes related to ASATE. The advantage with expanding the ASATE system to include satellites is the low cost and proven capabilities of the ASATE system.

Overall the development and usage of ASATE has provided significant benefit for relatively little cost. For academic institutions a simulation environment like ASATE has applications to both the research and teaching sides of the institution. The low cost and incorporation of hardware in the loop

makes system testing simple and effective. The visualisation and real time nature of ASATE provides a powerful design and teaching tool for control systems.

Acknowledgments

This work was carried out in the Cooperative Research Centre for Satellite Systems with financial support from the Commonwealth of Australia through the CRC Program.
I would also like to acknowledge James Hutchins for the work he performed with me in 2001 to design and implement the first version of ASATE.

References

[1] J. Hutchins, "QUAV Flight Management System," in *School of Electrical and Electronic Systems Engineering*. Brisbane: Queensland University of Technology, 2001.
[2] I. McManus, "QUAV Flight Control System," in *School of Electrical and Electronic Systems Engineering*. Brisbane: Queensland University of Technology, 2001.
[3] R. R. Costa, J. A. Silva, S. F. Wu, Q. P. Chu, and J. A. Mulder, "Atmospheric Re-entry Modeling and Simulation: Application to the Crew Return Vehicle," presented at AIAA Modeling and Simulation Technologies Conference and Exhibit, Denver, Colorado, 2000.
[4] S.-F. Wu, R. R. Costa, Q.-P. Chu, J. A. Mulder, and G. Ortega, "Nonlinear dynamic modeling and simulation of an atmospheric re-entry spacecraft," *Aerospace Science and Technology*, vol. 5, pp. 365-381, 2001.
[5] Unmanned Dynamics LLC, "Aerosim Blockset Version 1.01 User's Guide," Unmanned Dynamics LLC, Hood River, OR 2002.
[6] I. McManus and R. Walker, "Intelligent Agents "Hardware in the Loop" Simulator for UAV and Spacecraft Mission," presented at Cooperative Research Centre for Satellite Systems Conference, Adelaide, SA, 2003.
[7] I. McManus and M. Keating, "QUAV Project, Aircraft Simulation And Testing Environment v3, User's Manual," Queensland University of Technology, Brisbane QTAV-TL-STL-UM-0001-1, November 27 2002.
[8] I. McManus and M. Keating, "QUAV Project, Aircraft Simulation And Testing Environment v3, Control Systems Tutorial," Queensland University of Technology, Brisbane QTAV-TL-STL-UM-0002-1, November 27 2002.